Recursive Context-Aware Lexical Simplification

Sian Gooding
Dept of Computer Science and Technology
University of Cambridge
shg36@cam.ac.uk

Ekaterina Kochmar
ALTA Institute
Unversity of Cambridge
ek358@cam.ac.uk

Abstract

This paper presents a novel architecture for recursive context-aware lexical simplification, REC-LS, that is capable of (1) making use of the wider context when detecting the words in need of simplification and suggesting alternatives, and (2) taking previous simplification steps into account. We show that our system outputs lexical simplifications that are grammatically correct and semantically appropriate, and outperforms the current state-of-the-art systems in lexical simplification.

1 Introduction

Text simplification (TS) is aimed at reducing the reading and grammatical complexity of text while retaining the meaning and grammaticality (Chandrasekar and Bangalore, 1997). This is usually achieved by a series of transformations at the lexical and syntactic level. A number of systems in the recent years have approached this task in an integral manner (Zhu et al., 2010; Kauchak, 2013; Zhang and Lapata, 2017). Such comprehensive systems can perform a number of simplification operations at once, but the results are sometimes ungrammatical and meaning can be changed, arguably making the original text less clear and more complex (Siddharthan, 2014).

In this paper, we assume the lexical and syntactic components of a TS system to be independent and complementary to each other, and focus on the *lexical simplification* (LS) component for a number of reasons. First, it has been shown that *lexical simplification* techniques positively impact the readability of text and improve reader understanding and information retention (Leroy et al., 2012). Secondly, it has been argued that a large number of people with reading difficulties, including those with disabilities, low-literacy, non-native backgrounds or non-expert knowledge benefit from LS (Xu et al., 2015). For instance, James (1998) shows that vocabulary of the non-native language plays central role in second language acquisition. As we aim to make information more accessible to such readers, the quality of the simplified text is of a paramount importance.

We stress the importance of three key points of quality assessment in LS: it is vital for the simplified text to be *of a lower complexity*, while being *semantically equivalent* to the original, and *grammatically correct*. Context plays a central role in fulfilling these requirements. For instance, consider the different uses of *situation* in the complex word dataset collected by Yimam et al. (2017):

(1) the gravity of the **economic *situation***

(2) the *situation* has remained unchanged

This example demonstrates two types of contextual effects: first of all, in (1) both *economic* and *situation* are marked as complex in mutual context, but when *situation* occurs in a different context it is not annotated as complex. Secondly, this example illustrates the impact of the context on the choice of the appropriate substitution: substituting *climate* for *situation* will work for (1), but will result in a semantically different expression in (2).

In addition, we argue that as word complexity depends on context, the order and choice of applied simplifications matters. For instance, consider the following simplifications for the sentence '*This is a problem in the contemporary world*':

(3) This is a problem in the *modern earth*

(4) This is a problem in the *modern* world

Example (3) shows an output of a system that tries to simplify all words, which results in a nonsensical sentence, while (4) exemplifies the result of recursive word replacement in context.

Context effects in LS have not been thoroughly investigated before. In this paper, we introduce a

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 4853–4863,
Hong Kong, China, November 3–7, 2019.

novel approach to LS that addresses these issues. In particular, we make the following contributions:

1. As each simplification step changes the complexity of the output sentence, our LS algorithm applies simplification *recursively taking word complexity in context into account*;
2. To ensure grammaticality and meaning equivalence to the input in the output, our algorithm *takes context both at the complex word identification and substitution selection steps into account*. We use a novel sequence labelling component for the former step, and assess semantic equivalence at the latter using deep contextualized word representations provided by ELMo (Peters et al., 2018).
3. To facilitate reproducibility, we release the code and the output of our system at `github.com/siangooding/lexical_simplification`.

2 Previous work

2.1 Approaches

Early approaches to TS have mostly relied on rule-based systems (Carroll et al., 1998; Canning et al., 2000; Siddharthan, 2006), with many of the earlier systems prioritising syntactic operations, such as sentence splitting, deletion or reordering. Some work combined lexical simplification with syntactic operations (Zhu et al., 2010; Coster and Kauchak, 2011a; Kauchak, 2013). The availability of parallel corpora of "normal" and simplified text has inspired a number of approaches that treated TS as a monolingual machine translation problem (Zhu et al., 2010; Coster and Kauchak, 2011a,b) or allowed the researchers to apply language modelling (Kauchak, 2013).

Building on this line of work, Zhang and Lapata (2017) combine a novel sequence-to-sequence encoder-decoder model with a deep reinforcement learning framework that rewards the system for providing simple, fluent output, similar in meaning to the input. However, one of the main challenges for such comprehensive end-to-end systems is the ability to address specific types of errors independently. For instance, the DRESS-LS model (Zhang and Lapata, 2017) sometimes changes the meaning of the input to the opposite as in "*Inspections, she said,* **rarely** *cost more than $ 1,400*" → "*Inspections, she said,* **often** *cost more than $ 1,400*", or produces nonsensical output as in "*Archaeologists digging on the* **grounds**" → *Archaeologists digging on the* **zebras**".

A number of approaches focused on generation and assessment of *lexical simplification* (Yatskar et al., 2010; Biran et al., 2011; Horn et al., 2014; Glavaš and Štajner, 2015). Paetzold and Specia (2016a) note the lack of consistent evaluation for text simplification, and in particular lexical simplification, and introduce an evaluation dataset BENCHLS, on which they perform benchmarking of several LS systems. They argue that lexical simplification consists of a number of steps, including substitution generation, substitution selection and ranking. They show that the unsupervised system of Paetzold and Specia (2016b) outperforms a range of other systems in all steps, with the only exception of the feature-based system by Horn et al. (2014), which performs better in terms of precision in a round-trip system evaluation.

Finally, Shardlow (2013) introduced complex word identification (CWI) as the first step in an LS pipeline to detect words within text that require simplification. He showed that systems' performance on this stage is crucial for the overall performance, as low recall of this component might result in an overly difficult text with many missed complex words, while low precision might result in meaning distortions. The recent shared task on CWI shows that most systems rely on classification approaches using features that pertain to individual words, not taking wider context into account (Yimam et al., 2018).

2.2 Datasets

Until recently, parallel Wikipedia and Simple Wikipedia datasets have been the most widely used data for training and evaluating TS systems (Zhu et al., 2010; Yatskar et al., 2010; Coster and Kauchak, 2011a,b; Biran et al., 2011; Kauchak, 2013; Horn et al., 2014). Wikipedia allows free access to large quantities of data, and Simple Wikipedia represents a simplified version of original articles that uses simpler vocabulary and syntactic structures (Coster and Kauchak, 2011a). Researchers in the past applied monolingual alignment techniques to construct parallel versions of the two Wikipedias and learn the transformations from these parallel versions using such tools as GIZA++ (Och and Ney, 2000).

Despite the popularity of the Wikipedia-based

datasets for TS research, Xu et al. (2015) argue that focusing on Wikipedia limits simplification research and propose using a dataset based on news articles. They use a dataset from Newsela as an example, where the texts are simplified by professional editors at 4 levels of simplificity in accordance with the grade levels defined by the Common Core Standards (Porter et al., 2011).

In their benchmarking study, Paetzold and Specia (2016a) introduce an evaluation dataset BENCHLS that combines two previously released datasets for TS – LexMTurk (Horn et al., 2014) and LSeval (De Belder and Moens, 2012). This dataset contains 929 instances with an original sentence, a target complex word, and several candidate substitutions ranked by English speakers from the U.S. according to their simplicity. Paetzold and Specia (2016a) additionally filter out misspelled candidates and inflect all candidates to the grammatical form of the target word. The dataset contains an average of 7.37 candidate substitutions per complex word.

Finally, the CEFR-LS dataset (Uchida et al., 2018) contains simplifications, which not only represent a semantically good fit and are grammatically correct, but are also at different levels of simplicity annotated with respect to non-native speakers. Simpler candidates at lower levels of language proficiency (A1-B1) according to the Common European Framework of Reference for Languages (CEFR) (Council of Europe, 2011) are provided for the original words that are at higher levels of language proficiency (B2-C2). The dataset contains 406 target words and 4912 possible substitutions. Unlike BENCHLS, substitute candidates within this dataset may not be correct for the given context, and if a substitute candidate is not appropriate in context it is labelled as such. The dataset contains an average of 2.35 candidate substitutions per complex word.

3 Data

In this study we implement an LS system that includes complex word identification, substitute generation, filtering and ranking as steps in a simplification pipeline. To train our CWI system, we use the CWI 2018 shared task dataset (Yimam et al., 2017), which contains texts on three genres – professionally written NEWS, amateurishly written WIKINEWS, and WIKIPEDIA articles. The words in the dataset are annotated as complex or not by 10 native and 10 non-native speakers of English.

We evaluate each step of the LS pipeline on two datasets – BENCHLS and CEFR-LS. We select the BENCHLS dataset because it contains multiple simplification alternatives ranked with respect to their simplicity by a number of human annotators. The CEFR-LS dataset is a useful resource for evaluation because, in addition to contextually suitable, grammatically correct, simpler alternatives, it also contains substitution candidates that do not fit the context. Furthermore, this dataset is aimed at non-native speakers of English which we view as the future target group for our LS system. For further details on datasets collection and annotation, we refer the readers to the original papers.

Finally, we evaluate our LS system in an end-to-end manner and compare its performance to that of the current state-of-the-art systems, including those reported in Paetzold and Specia (2016a) and the DRESS-LS system of Zhang and Lapata (2017). In contrast to Zhang and Lapata (2017), we perform lexical simplification only. For fair comparison of the two systems, we extract only lexical simplifications from the parallel "normal" to simplified versions of the data used in Zhang and Lapata (2017), as well as from the original "normal" text and the DRESS-LS system output.

To extract the lexical transformations from the data, we use GIZA++ (Och and Ney, 2000) similarly to previous research (Coster and Kauchak, 2011b; Xu et al., 2015). Following Horn et al. (2014), we extract the examples that constitute one-to-one word correspondences between the two sides identified by the automatically induced word alignment, where the part-of-speech tag of the two words is the same while the lemmas are different. In addition, we filter out the instances involving modification of stopwords on the original, "normal" side as well as rewrites involving proper nouns. All preprocessing steps for our algorithm are performed using the RASP parser (Briscoe et al., 2006).

4 Lexical Simplification Pipeline

Lexical simplification can be viewed as a multi-stage process involving complex word identification, substitute generation, filtering and substitute ranking. In this section we outline each step of the simplification process as presented in Figure 1.

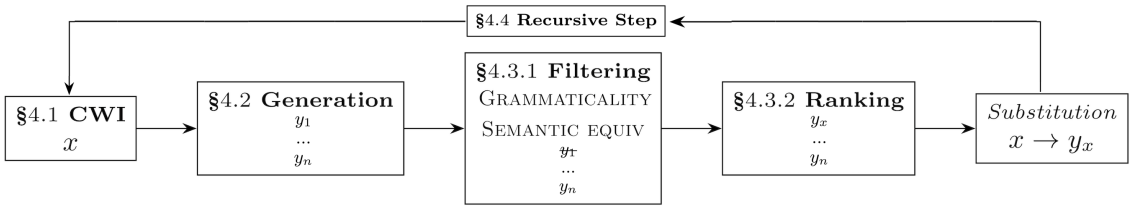

Figure 1: Recursive Simplification (REC-LS) pipeline: x is a complex word and y_i are the generated substitutes

4.1 Complex Word Identification (CWI)

The first stage of the algorithm is complex word identification (CWI), which aims to identify which words should be simplified within the text and, thus, allows for a personalisable and targeted approach to LS. Furthermore, it helps to reduce the number of unnecessary and potentially "harmful" simplifications performed by a 'simplify all words' approach. To train and test our CWI component we use the dataset by Yimam et al. (2017).

Word complexity depends on the surrounding context: for instance, between 3% and 10% of lexical items (depending on genre) in the Yimam et al. (2017) dataset receive different annotations in different contexts. However, most CWI systems to date have approached this task on an individual word basis (Yimam et al., 2018). For instance, the 2018 CWI shared task winning system CAMB uses a classification-based approach with 27 word-level features (Gooding and Kochmar, 2018). As context impacts the perceived complexity of text, we argue that CWI should be framed as a sequence labelling task and propose a novel architecture SEQ based on this idea (Gooding and Kochmar, 2019). We extend the implementation of a sequence labeller by Rei (2017),[1] which achieves state-of-the-art results on a number of NLP tasks. The design of this architecture is highly suitable for CWI as: (1) it uses bi-directional long short-term memory units (BiLSTM) (Hochreiter and Schmidhuber, 1997), which allow the system to learn about both the left and right context of a target word; (2) the context is combined with both word and character-level representations (Rei et al., 2016) which helps capture complexity due to rare character sequences as well as morphological structure; (3) this architecture uses a language modelling objective, which enables the model to take one of the highly informative complexity factors of word frequency into account.

[1] https://github.com/marekrei/sequence-labeler

Test Set	Macro F-Score	
	CAMB	SEQ
NEWS	0.8633	**0.8763**
WIKINEWS	0.8317	**0.8540**
WIKIPEDIA	0.7780	**0.8140**

Table 1: SEQ vs. CAMB system results on CWI

DATASET	Precision	Recall	F-Score
CEFR-LS	0.9327	0.7936	0.8575
BENCHLS	0.5000	0.4025	0.4460

Table 2: SEQ results for CWI on the two datasets

SEQ uses 300-dimensional GloVe embeddings as word representations (Pennington et al., 2014). The model is trained to predict the binary complexity of words as annotated in the dataset of Yimam et al. (2017). Training is performed over 20 iterations on randomly shuffled sentences from all genres included within the dataset. To test this novel architecture on CWI, we apply it to the CWI 2018 shared task test data (Yimam et al., 2018) and compare the results to the current state-of-the-art (SOTA) CAMB system. Table 1 shows that the SEQ model outperforms the current SOTA system on all three text genres for binary CWI (statistically significant using McNemar's test, p=0.0016, χ^2=9.95).

The proposed SEQ model (Gooding and Kochmar, 2019) has a number of additional advantages: it takes context into account, helps avoid the necessity of extensive feature engineering relying on word embeddings as the only input information at run time, and generalises well across all three datasets. To further assess generalisability of the model, we test it on CEFR-LS, as well as BENCHLS for consistency (see Table 2). However, as the words in BENCHLS dataset were selected at random rather than according to their complexity (Paetzold and Specia, 2016a), the results as expected are lower.

4.1.1 Lexical Complexity Threshold

The SEQ system labels each word with a lexical complexity score. This score represents the likelihood (p) of each word belonging to the complex class. Whether a word is considered as complex is set according to a predefined threshold for p, which allows the 'aggressiveness' of the algorithm to be tailored according to the application. For instance, consider the following example:

(5) I believe that ignoring public opinion ***discredits***$_{0.89}$ the *authorities*$_{0.79}$ and ***destabilises***$_{0.84}$ the situation.

The SEQ labeller assigns complexity probability p>0.80 to the words ***discredits*** and ***destabilises***, whereas for *authorities* p=0.79. If the complexity threshold in the simplification pipeline is set to 0.80, the word *authorities* would not be a candidate for lexical simplification. Our simplification pipeline will attempt to simplify words one at a time, starting with the highest p values above this predefined threshold.

4.2 Substitute Generation

Substitute generation refers to the process of generating candidates that can be used as simpler alternatives to the target word. The goal of the system at this stage is to generate a diverse set of potential substitutions that can be filtered and ranked according to different criteria at later steps. We note that the use of multiple resources for substitute generation at this step is crucial as each individual resource has only a limited coverage. To this end, we first use traditional approaches, which do not take the context of the target word or the simplicity of substitutes into account. In summary, we rely on the following resources:

- we extract synonyms from WordNet (Fellbaum, 2012), following Devlin (1998);
- we use Big Huge Thesaurus[2] to find candidates by querying for the word form and lemma;
- following Glavaš and Štajner (2015), we use word embeddings to create substitutions. Synonyms are identified by finding 10 words with the highest cosine similarity to the target word in the vector space. These are obtained for both GloVe (Pennington et al., 2014) and Word2Vec embeddings (Mikolov et al., 2013), using the Gensim library.[3]

The candidate lemmas retrieved using these resources are added to the candidate substitutions set. To match the grammatical form of the original word, we convert all candidates to the appropriate word forms using NodeBox English Linguistics library.[4]

O	Water **engulfed**$_{0.89}$ Beringia.
S_1	Water *overwhelmed*$_{0.78}$ Beringia.
S_2	Water *flooded*$_{0.39}$ Beringia.

Table 3: Contextual simplicity scores for 2 possible alternatives (S_1-S_2) for the original O

4.3 Substitution Filtering and Ranking

Following substitute generation, the next step in the LS pipeline is to choose one of the generated candidates to replace the original target word. This is done by filtering and ranking the candidates using a set of criteria, and choosing the top candidate. Previous work has framed this task as ranking words according to simplicity (Paetzold and Specia, 2016a). However, a system that is only aimed at selecting the simplest candidate may return a substitution that does not fit the surrounding context, is ungrammatical or not semantically equivalent to the original. Therefore, we consider all three aspects of high-quality substitution selection: *contextual simplicity*, *semantic equivalence* and *grammaticality*, which are outlined below.

Contextual Simplicity: The complexity of each candidate is calculated using our sequential CWI model from Section 4.1. We calculate the complexity for each substitution within context and refer to this as the simplicity score S. Table 3 shows 2 possible substitutions for *engulfed*, with their respective simplicity scores.

Contextual Semantic Equivalence: To assess whether a substitution is semantically equivalent to the original we use ELMo embeddings (Peters et al., 2018), which to the best of our knowledge have not been used for LS before. ELMo provides deep contextualized word representations, which are learned from the internal states of a deep bidirectional language model. As a result, these embeddings are able to model complex syntactic and

[2] https://words.bighugelabs.com

[3] https://github.com/RaRe-Technologies/gensim

[4] https://www.nodebox.net/code/index.php/Linguistics

O	This **illustrates** how sociologists [...]
S_1	This *demonstrates*$_{0.10}$ how sociologists [...]
S_2	This *shows*$_{0.16}$ how sociologists [...]
S_3	This *draws*$_{0.44}$ how sociologists [...]

Table 4: Contextual semantic equivalence scores for 3 candidate substitutions (S_1-S_3) for the original O

O	Oak is strong and also *gives* shade
g.c.	Oak *gives* shade

Table 5: Original sentence O and the grammatical context (g.c.) used to calculate C_G for the word *gives*

semantic characteristics of word use, as well as how these word uses vary across linguistic contexts. In addition, contextualised embeddings help filter out antonyms that might be included in the set of potential substitutes by the previous step.

The similarity of words in a given sentential context is calculated using the ELMo embeddings associated with the target word and each of the substitutes. The contextual similarity score (C_S) is calculated by taking the cosine distance between the ELMo word vectors associated with the original word v_o and the substitute v_s. This allows us to reason about the likelihood that the substitution is semantically equivalent to the original in the given context: the *smaller* the distance the *more likely* this word fits the context. Table 4 provides some examples for this score.

This technique works particularly well when the target word appears in the immediate context of the words grammatically related to it, for example when a verb is surrounded by its subject and object. To counteract the effect of long range dependencies in sentences, we constrain the context to the grammatical dependents and calculate a second contextual score C_G using cosine distance. Table 5 provides an example of a full sentence O and the grammatical context for the word *gives*.

Grammaticality: To assess whether a substitution results in a grammatically correct sentence, we calculate bigram frequencies of the candidate substitute and one word to the left and to the right of it using the 520 million word COCA corpus (Davies, 2014). If either of these frequencies equals 0, it is assumed that the substitute is not a valid grammatical fit or is extremely rare, making it a poor candidate for simplification. Table 6 shows one such example.

	left	*right*
He was **capable** of [...]		
bigram	was capable	capable of
freq	778	12341
He was **able** of [...]		
bigram	was able	able of
freq	10281	0

Table 6: Bigram frequencies for left and right contexts of a candidate substitute

Here, we use the bigram frequencies as a proxy for grammaticality: although "able" and "capable" are semantically similar, it is the grammatical constraints, captured by the bigram frequencies, that rule one of the alternatives out.

4.3.1 Threshold-based filtering

Threshold-based filtering is performed by removing all substitutes that are unlikely to be grammatical or do not fit the target context. First, substitutes are removed from consideration if their right or left bigram frequency equals 0. Then, we remove substitutes if either of their contextual ELMo scores is below a given threshold t ($C_S \vee C_G < t$), as this implies the substitute is not equivalent in this context. We test our filtering approach on the CEFR-LS dataset as it contains annotations for contextually suitable (value 1) as well as unsuitable (value 0) substitutions. We empirically find the optimal threshold value on the CEFR-LS dataset to be 0.175. With this optimal value, we can identify contextually suitable and grammatically correct substitutes on CEFR-LS with the *precision* of 0.7968, *recall* of 0.8081 and F_1=0.8014. As we show in Section 5, this filtering technique generalises well to other datasets.

4.3.2 Ranking Algorithm

Once filtering has been performed we then rank substitutes. In order to rank substitutes, the simplicity and contextual semantic equivalence scores are combined to produce an overall suitability score. We evaluate our ranking techniques on the BENCHLS and CEFR-LS datasets.

We perform ranking using the sum of the contextual simplicity score (S) and the average contextual semantic equivalence score $\bar{C} = avg(C_S, C_G)$, and evaluate the results using the *TRank-at-n* measure introduced in Paetzold and Specia (2016a), which estimates the proportion of times a candidate with a gold-standard rank

BENCHLS		n=1	n=2	n=3	MRR
$full_{(929)}$	S	0.4974	0.7381	0.8899	0.6648
	$\bar{C}$	0.3509	0.5885	0.7877	0.5998
	S+$\bar{C}$	**0.5602**	**0.8064**	**0.9428**	**0.7219**
$test_{(464)}$	S	0.5839	0.7546	0.9302	0.7083
	$\bar{C}$	0.4086	0.7142	0.8950	0.6563
	S+$\bar{C}$	**0.6774**	**0.7857**	**0.9308**	**0.8218**
	P&S	0.4841	0.5596	0.7004	0.6615

Table 7: Ranking results on BENCHLS dataset

CEFR-LS	n=1	n=2	n=3	MRR
S	0.2166	0.3357	0.5271	0.4182
$\bar{C}$	0.3754	**0.6137**	**0.7689**	0.5857
S+$\bar{C}$	**0.4624**	0.5906	0.7314	**0.6582**
P&S	0.2129	0.4339	0.5760	0.4736

Table 8: Ranking results on CEFR-LS dataset

$r \leq n$ is ranked first by the system. For instance, our best performing ranking technique on the full BENCHLS dataset for n=1 is based on the combination of S+$\bar{C}$ scores and achieves *TRank-at-n*=0.5602. This means that for approximately 56% of the test instances the top ranked substitute in the gold standard is correctly ranked first. In addition, we report the *mean reciprocal rank (MRR)* (Voorhees, 1999), which takes into account the rank of the substitutes proposed by each ranking technique.

We report the results on the full BENCHLS dataset in the upper half of Table 7. In the lower half, we compare our results on the test set of 464 instances to those running the Paetzold and Specia (2016a) system (P&S) on the same test splits. Since the P&S system was trained on half of BENCHLS we cannot run it on the full dataset. Table 7 shows that ranking with S+$\bar{C}$ works best according to all measures and across both sets.

Table 8 reports the ranking results on the CEFR-LS data. We observe a decrease in performance, however this is expected: as all substitutes within the BENCHLS dataset are valid, ranking by simplicity is more informative; in contrast, the CEFR-LS dataset contains irrelevant substitutes, so contextual fit has more pronounced effects.

4.4 Recursive Step

Following CWI, substitute generation, filtering and substitute ranking steps, the system is able to perform a simplification. As outlined in Section 4.1.1, our system attempts to simplify one word at a time, starting with the word considered most complex. Since word complexity depends on the context, each individual lexical simplification made to a sentence has a subsequent impact on the perceived complexity of the surrounding words. Such sequential effects cannot be modelled by systems that apply several simplification steps at once. For instance, when considering examples (6) and (7) we see that the word *situation* is given a high likelihood of being complex in the context of another complex word *hazardous*. However, once *dangerous* is substituted for *hazardous*, the subsequent complexity of *situation* is reduced as well.

(6) It was a **hazardous**$_{0.90}$ **situation**$_{0.87}$

(7) It was a dangerous$_{0.30}$ situation$_{0.35}$

Our lexical simplification algorithm is applied to a sentence recursively: it starts by identifying and simplifying the most complex words in the sentence. Once the simplification is applied in step n, the algorithm reassesses word complexity in step n+1, which, in light of the simplifications applied in previous steps, might have changed. This prevents unnecessarily simplifying *situation* in example (7), which is no longer necessary after simplifying *hazardous* → *dangerous*. The simplification algorithm stops when there are no words with the complexity score above the predefined threshold left within the sentence, set to 0.5.

Table 9 exemplifies the benefit of using the recursive simplification approach REC-LS. We see that both the 'Simplify CW', aimed at individually identified complex words, and 'Simplify all' approaches result in unnecessary simplifications, whilst REC-LS stops when it recognises that the surrounding words are no longer complex.

Original	*prepare*$_{0.32}$ *for a* *hazardous*$_{0.81}$ *journey*$_{0.52}$
REC-LS	prepare for a **dangerous**$_{0.35}$ journey$_{0.45}$
Simplify CW	prepare for a **dangerous trip**
Simplify all	**arrange** for a **dangerous trip**

Table 9: Recursive simplification compared with simplifying only complex words and simplifying all words

We note that the only consequence of performing simplification recursively is that fewer words, or potentially different words, are simplified, and highlight that it does not lead to any error propagation. Table 10 presents the entire REC-LS algorithm including the recursive simplification step.

5 End-to-end System Performance

Finally, we test the full LS pipeline, visualised in Figure 1, on the BENCHLS and CEFR-LS

Result: Lexically simplified sentence
$S \leftarrow$ Input Sentence
$c \leftarrow$ Complexity threshold
REC-LS$(S, c, ignore_list)$
while *complexity(S)* $> c$ **do**
  complex_words $\leftarrow$ CWI(S) - *ignore_list*
  SORT(complex_words)
  for *word in complex_words* **do**
    subs $\leftarrow$ SUB_GENERATION$(word)$;
    subs $\leftarrow$ MORPH$(substitutes)$;
    subs $\leftarrow$ SUB_RANK$(substitutes)$;
    top_substitute $\leftarrow head(substitutes)$;
    if *top_substitute == word or []* **then**
      ignore_list.add(*word*)
      REC-LS$(S, c, ignore_list)$
    else
      REPLACE$(S, word, top_substitute)$
      REC-LS$(S, c, ignore_list)$
  end
end

Table 10: REC-LS Algorithm Steps

(4)	Check if complexity of any word in S is above the threshold c
(5)	Check sentence for complex words minus those which are in *ignore_list*
(6)	Sort complex words according to confidence score
(8)	Generate substitutes for the top complex word
(9)	Convert substitutions to correct word form (e.g., tense for verbs)
(10)	Rank substitutes according to grammaticality, contextual semantic equivalence and simplicity
(11)	Take the top substitute
(13)	If no appropriate substitutes found, or best fit is original word, then retain original word and add word to *ignore_list*
(15)	Otherwise replace word and call function with new sentence

datasets. The CWI step of our recursive LS algorithm identifies 77% of the target complex words in the BENCHLS and 86% of the target complex words in the CEFR-LS dataset. Next, the substitution generation and ranking produce lists of simplification candidates. We evaluate these steps using *precision* and measuring the percentage of times that our system ranks one of the gold standard candidates as its top choice for substitution.

The best precision P=0.7945 on BENCHLS is achieved with threshold-based filtering of the unsuitable candidates and ranking of the candidates according to the combination of contextual simplicity (S) and contextual fit ($\bar{C}$) scores. We note that this result outperforms the previous SOTA system by Horn et al. (2014), which is reported to have P=0.5460 (Paetzold and Specia, 2016a), by a large margin.

With a similar approach, we achieve P=0.4628 on the CEFR-LS, and to the best of our knowledge, this is the first time an LS system is benchmarked on this dataset. We note that the precision on this dataset is lower, which can be attributed to the smaller set of gold standard substitutes per word (an average of 2.35 in CEFR-LS vs 7.37 in BENCHLS). We also note that in a number of cases our system generates valid substitutes, which are not included in the gold standard. For example, *wealthy* → *rich* in "could participate in government just as **wealthy** men could". We present more examples of such cases in the Appendix.

We also run our entire recursive simplification system REC-LS on the three simplification datasets: WikiSmall, WikiLarge and Newsela, used in Zhang and Lapata (2017). We then compare the results to the SOTA simplification systems DRESS-LS by Zhang and Lapata (2017), which contains a specialised LS model, and the P&S simplification system (Paetzold and Specia, 2016a). As DRESS-LS is trained using the above datasets, we test using the 649 sentences that are reserved for testing only. Simplifications are assessed using the gold standard for lexical substitutions. For each dataset, the number of lexical simplifications performed by the systems is recorded. We then compare the simplifications performed by the system with the gold standard simplifications and calculate the *recall*, *precision* and *correct* proportion as follows:

- For *recall* we estimate the number of simplifications present in the gold standard $|G|$, and the total number produced by the system S, which are in the gold standard $|S \cap G|$. Recall is then calculated as $|S \cap G| / |G|$.
- For *precision*, we identify the proportion of simplifications out of the total $|S|$, which are also in the gold standard: $|S \cap G| / |S|$.

	Newsela			*WikiSmall*			*WikiLarge*		
	DRESS-LS	P&S	REC-LS	DRESS-LS	P&S	REC-LS	DRESS-LS	P&S	REC-LS
Precision	0.1673	0.0836	**0.3000**	0.0000	0.0602	**0.1275**	0.2258	0.0697	**0.2778**
Recall	0.2389	**0.9721**	0.2393	0.0000	**0.9743**	0.4815	0.0004	**0.9763**	0.1856
Correct	0.3256	0.0402	**0.5238**	0.0000	0.0526	**0.4615**	**0.4286**	0.0787	0.4001

Table 11: Lexical substitution results for DRESS-LS, P&S and our system (REC-LS) on three genres

- Finally, *correct* stands for the proportion of instances where the top lexical substitution returned by the system is exactly the same as the gold standard one.

The results show that the recall for REC-LS is higher than that of DRESS-LS across all datasets, while the P&S system performs the best in terms of recall as it applies a 'simplify all' approach. The difference is especially pronounced in the WikiSmall dataset, where the DRESS-LS system does not perform any required lexical substitutions. REC-LS outperforms both DRESS-LS and P&S systems across all datasets in terms of precision, which indicates that the CWI stage of the algorithm is able to narrow down simplifications to relevant words. Finally, the three systems show different results in terms of correct proportion: REC-LS outperforms other systems on Newsela and WikiSmall, and DRESS-LS has a higher correct proportion on WikiLarge.

Finally, we note that there are many instances where the REC-LS system performs substitutions with valid alternatives that are not contained within the gold standard. For instance, consider the word "*separated*" in the following context:

(8) The island chain forms part of the Hebrides, *separated* from the Scottish mainland.

The gold standard substitution suggested by the human annotators in this context is "demarcated", whereas REC-LS substitutes this word with "split". We argue that, in this context, "split" would be both grammatically correct and semantically appropriate, while also being simpler than the original word and, arguably, the substitute suggested by the human annotators. However, as it is not an exact match with the gold standard substitution in the dataset, cases like this negatively impact precision of our system. We provide more examples of the system's output and compare the simplifications from the three systems in the Appendix.

6 Conclusions

In this paper, we have presented a novel recursive context-aware lexical simplification architecture REC-LS.[5] We summarise the main contributions of this work as follows:

1. our model takes the surrounding context into account at the complex word identification step, using a novel sequence labelling approach;
2. it filters and ranks the substitution alternatives taking contextual fit and grammaticality of the output into account;
3. the novel architecture performs simplification recursively and can be adapted to different levels of language complexity in the future;
4. as a result, REC-LS outperforms the current state-of-the-art systems in lexical simplification on various datasets.

We note that REC-LS can be used as a stand-alone lexical simplification system as well as an LS component as part of an integral TS system.

In this paper, we focus on intrinsic evaluation of our system, and we show that it outperforms SOTA in terms of the widely accepted set of measures, which we believe to be important for benchmarking purposes. In the future, we plan to extend this work with a syntactic simplification component and test the architecture extrinsically with non-native readers.

Acknowledgments

We thank Cambridge English for supporting this research via the ALTA Institute. We are grateful to the anonymous reviewers for their valuable feedback. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.

[5]The code and output is available on github.com/siangooding/lexical_simplification

References

Or Biran, Samuel Brody, and Noemie Elhadad. 2011. Putting it Simply: a Context-Aware Approach to Lexical Simplification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 496–501.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 77–80. Association for Computational Linguistics.

Yvonne Canning, John Tait, Jackie Archibald, and Ros Crawley. 2000. Cohesive generation of syntactically simplified newspaper text. In *Proceedings of the Third International Workshop on Text, Speech and Dialogue (TDS '00)*, pages 145–150.

John Carroll, Gido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. 1998. Practical simplification of English newspaper text to assist aphasic readers. In *Proceedings of AAAI Workshop on Integrating AI and Assistive Technology*, pages 7–10.

Raman Chandrasekar and Srinivas Bangalore. 1997. Automatic induction of rules for text simplification. *Knowledge Based Systems*, 10:183–190.

William Coster and David Kauchak. 2011a. Learning to Simplify Sentences Using Wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 1–9.

William Coster and David Kauchak. 2011b. Simple English Wikipedia: A New Text Simplification Task. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 665–669.

Mark Davies. 2014. N-grams data from the Corpus of Contemporary American English (COCA).

Jan De Belder and Marie-Francine Moens. 2012. A dataset for the evaluation of lexical simplification. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 426–437. Springer.

Siobhan Devlin. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic databases*.

Council of Europe Council of Europe. 2011. Common European Framework of Reference for Languages: Learning, Teaching, Assessment.

Christiane Fellbaum. 2012. WordNet. *The Encyclopedia of Applied Linguistics*.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: do we need simplified corpora? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 63–68.

Sian Gooding and Ekaterina Kochmar. 2018. CAMB at CWI Shared Task 2018: Complex Word Identification with Ensemble-Based Voting. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194.

Sian Gooding and Ekaterina Kochmar. 2019. Complex Word Identification as a Sequence Labelling Task. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1148–1153.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a Lexical Simplifier Using Wikipedia. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 458–463.

Carl James. 1998. *Errors in Language Learning and Use: Exploring Error Analysis*. London: Longman.

David Kauchak. 2013. Improving Text Simplification Language Modeling Using Unsimplified Text Data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1537–1546.

Gondy Leroy, James E. Endicott, Obay Mouradi, David Kauchak, and Melissa L. Just. 2012. Improving perceived and actual text difficulty for health information consumers using semi-automated methods. In *Proceedings of the American Medical Informatics Association (AMIA) Fall Symposium*, pages 522–531.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Franz Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL '00)*, pages 440–447.

Gustavo Paetzold and Lucia Specia. 2016a. Benchmarking lexical simplification systems. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 3074–3080, Paris, France. European Language Resources Association (ELRA).

Gustavo Paetzold and Lucia Specia. 2016b. Unsupervised lexical simplification for non-native speakers. In *Proceedings of The 30th AAAI*, pages 3761–3767.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT 2018*, pages 2227–2237.

Andrew Porter, Jennifer McMaken, Jun Hwang, and Rui Yang. 2011. Common Core Standards: The New U.S. Intended Curriculum. *Educational Researcher*, 40:103–116.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2121–2130.

Marek Rei, Gamal K.O. Crichton, and Sampo Pyysalo. 2016. Attending to characters in neural sequence labeling models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 309–318.

Matthew Shardlow. 2013. A Comparison of Techniques to Automatically Identify Complex Words. In *Proceedings of the Student Research Workshop at the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 103–109.

Advaith Siddharthan. 2006. Syntactic Simplification and Text Cohesion. *Research on Language and Computation*, 4:77–109.

Advaith Siddharthan. 2014. A survey of research on text simplification. *Special issue of International Journal of Applied Linguistics*, 165:259–298.

Satoru Uchida, Shohei Takada, and Yuki Arase. 2018. CEFR-based Lexical Simplification Dataset. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, pages 3254–3258.

Ellen M. Voorhees. 1999. TREC-8 Question Answering Track Report. In *Proceedings of the 8th Text Retrieval Conference*, pages 77–82.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in Current Text Simplification Research: New Data Can Help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: Unsupervised extraction of lexical simplifications from Wikipedia. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 365–368.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A Report on the Complex Word Identification Shared Task 2018. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017. CWIG3G2 - Complex Word Identification Task across Three Text Genres and Two User Groups. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 401–407.

Xingxing Zhang and Mirella Lapata. 2017. Sentence Simplification with Deep Reinforcement Learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A Monolingual Tree-based Translation Model for Sentence Simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361.